

# Informatik für Schüler, Foliensatz 22 Wiederholung Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal 30. April 2009

## Grundbegriffe der objektorientierten Programmierung

```
class Klassenname():
\{Attribut = Wert\}
{def Methode(self {, arq}):
   Anweisung
   {Anweisung}}
Objektname = Klassenname()
Objektname. Methode (Argumente_außer_self)
a = Objektname.Attribut
```

#### Was ist:

- ein Attribute, eine Methode, eine Klasse
- ein Objekt
- das Argument »self«?

### Welche Fehler enthält die Klassendefinition

```
class text():
# Attribute
L = []
 s=''
 def append(self, txt):
  s += txt
  def insert(self, txt, pos):
  self.s = self.s[:pos] + txt + self.s[pos:]
 def show():
  print s
```

■ Was ist an folgenden Anweisungen falsch? a=text a.append(self, 'Katze')

### Eine Zeichenfeldklasse

class ZF2D():

```
Attribute
     zf=:[]
     z = 0 \# Zeilenanzahl
     s = 0 # Spaltenanzahl
```

#### Methoden

■ Erweiterung der leeren Liste »zf« zu einer 2D-Liste (siehe Foliensatz 13, Folie 6):

```
def create(self, Zeilen, Spalten, Zeichen):
  self.s=Spalten
  self.z=Zeilen
  for idx in range(Zeilen):
   self.zf.append(Spalten*[Zeichen])
```

Umwandeln in eine Zeichenkette

```
def to_str(self):
  s = ''
  for Zeile in self.zf:
   for Zeichen in Zeile:
    s += Zeichen
   s += 'n'
  return<sup>1</sup> s
```

Anzeigen des Zeichenfelds

```
def show(self):
print '\nZeichenfeld ', self.s, 'x', self.z
print self.to_str()2
```

<sup>&</sup>lt;sup>1</sup>auch Methoden können einen Rückgabewert haben

<sup>&</sup>lt;sup>2</sup>Aufruf einer eigenen Methode der Klasse

# TU Clausthal

Zeichnen eines Punktes

```
def drawPoint(self, x, y, Zeichen):
 s = int^3(round^4(x))
z = int(round(y))
 if 0<=s and s<self.s and 0<=z and z<self.z:5
  self.zf[z][s] = Zeichen
```

<sup>&</sup>lt;sup>3</sup>Umwandlung in »int«, Feldindex muss Typ »int« haben

<sup>&</sup>lt;sup>4</sup>es soll auch float als Eingabe zugelassen sein; »float« muss auf ganze Zahl gerundet werden; konvertiert auch »int« nach »float«

<sup>&</sup>lt;sup>5</sup>Kontrolle, dass der Punkt im Zeichenfeld liegt

**Z**eichnen einer Linie mit den Endpunkten  $\mathbf{a} = (a_x, a_y)$  und  $\mathbf{b} = (b_x, b_y)$ 

```
def drawLine(self, ax, ay, bx, by, Zeichen):
  if abs(ax-bx)>abs(ay-by)^6:
   for x in range(int(round(ax)), int(round(bx))):
     y= (ay + (by-ay)/(float^7(bx)-ax)*(x-ax))
     self.drawPoint(x, y, Zeichen)
  else:
   for y in range(int(round(ay)), int(round(by)))8:
     x = (ax + (bx-ax)/(float(by)-ay)*(y-ay))
     self.drawPoint(x, y, Zeichen)
```

<sup>6</sup>Anstieg kleiner 1: Wiederhole von der Anfangs bis zur Endspalte: Berechne y

<sup>7</sup>Stellt sicher, dass der Anstieg als Gleitkommazahl berechnet wird, ganzzahlige Rechnung ergibt Null

 $^8{\rm Anstieg}$ größer 1: Wiederhole von der Anfangs- bis zur Endzeile: Berechne x

### Test der Klasse

```
a=ZF2D()
a.create(7, 40, '*')
a.drawPoint(1, 30, 'U')
a.drawLine(3, 2, 34, 8, 'x')
a.show()
```

Ausgabe

```
Zeichenfeld 7 x 40
*************************
************
*************
*************
*************
```



### Aufgabe 22.1: Linienmuster

Erzeugen Sie mit einem Objekt und den Methoden der Klasse ZS2D folgendes Muster:

```
Zeichenfeld 20 x 80
* * * * * * * * * * *
```

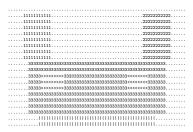
# TU Clausthal

- Zeichenfeld erzeugen
- Wiederhole für alle Linienanfangspunkte (auch die nicht sichtbaren)
  - Zeichne eine Linie mit dem Entpunkt in der untersten Zeile 20 Spalten weiter hinten
  - Zeichne eine Linie mit dem Entpunkt in der untersten Zeile 20 Spalten weiter vorn

### Aufgabe 22.2: Hasenbild

Entwickeln Sie für die Klasse ZF2D eine Methode def drawRect(self, ax, ay, bx, by, Zeichen): . . .

die eine Rechteck mit den Diagonalenenden  $\mathbf{a} = (a_x, a_y)$  und  $\mathbf{b} = (b_x, b_y)$  mit Zeichen füllt und schreiben Sie eine Testprogramm, dass wieder das bekannte Bild erzeugt:



### 22.3: Textfeld

- Erweitern Sie die Klassendefinition um eine Methode def drawText(self, x, y, Text) die ab der Position (x, y) einen Text in des Zeichenfeld schreibt.
- Schreiben Sie eine Testprogramm, das die Werte des Tupelst = ((1, 23), (2, 48), (3, 12345));

mit einer Schleife wie folgt tabellarisch darstellt:

Zeichenfeld	10	X	40

Eingabe	Ausgabe
1	23
2	48
3	12345